Notification Service Working using BullMQ

Overview

This document outlines the implementation of notification service using BullMQ for message queuing. The service is designed to handle different types of notifications, such as emails, Android notifications, iOS notifications, and web push notifications.

1. Queue Configuration

- **We can do two things here**:

1. Create **one single queue** named ***'notifications'*** which would be responsible for sending any notification related payloads.

2. Create **multiple queues** depending on the type of notification. For e.g. different queue for sending emails, different queue for sending app notifications and so on.

3.We can also segregate different notification operations logic by naming the job. So, let's say if we need to send notification for users to remind them to complete their profile, we can name the job as profile-remember, and the subscriber(worker) can have the logic written as per the job name.

2. Worker Configuration

- **We can do two things here**:

1. Create **one single worker** named ***'notifications'*** which would be responsible for sending notifications. In this case, **we can send in a *'type'* key** in the payload so that we can handle different types of notifications within one single Worker on the basis of a switch condition.

2. Create **multiple workers** depending on the type of notification. For e.g. different worker for sending emails, different worker for sending app notifications and so on.

3. **Notification Payloads**

- This is something which will differ as per the type of the notification that we have to send.
- Let's say we need to send android notifications the payload could be:
  Array<{
  deviceToken,
  title,
  message}>

- Or else we can keep **predefined messages** on the notification service itself rather than any other service being responsible to send the message string/template..
  For e..g let's say we are to send a welcome email to the users who just created their account, in that case the Worker can accept firstname,lastname and email of the user and we could have a message template on the notification service itself.

4. Sending Notifications

- We can create a **'sendNotification'** function to enqueue notification tasks for both single and bulk notifications.

5. Handling Errors

- Configure the worker to handle job failures and retries using BullMQ options and maybe set a backoff strategy as well.

P.S. : **Kindly let me know if anything is unclear or needs more explanation or any in depth explanation is required.**